Od dłuższego czasu staramy się przekonać Czytelników, że współczesne zastosowania komputerów osobistych odbiegają bardzo daleko od pierwotnej funkcji komputera (ang. compute — liczyć). Chociaż przy przetwarzaniu danych komputery faktycznie wykonują dużo obliczeń (stąd testy szybkości i dokładności dla komputerów) jest to jednak czynność zupełnie niedostrzegalna z poziomu użytkownika i równie mało interesująca jak procesy chemiczne w brzuchu konia pociągowego.

W poniższym tekście Jakub Tatarkiewicz opisuje kolejne „nieliczące" zastosowanie komputera, mimo że przeznaczone tym razem dla osób... intensywnie posługujących się matematyką. „Osobista" wersja programu MACSYMA rozprowadzana jest przez firmę MICROSOFT pod nazwą muMAT w cenie ok. 250 dolarów. Oferowana jest wersja dla komputerów 8-bitowych, pracująca pod kontrolą systemu operacyjnego CP/M, jak również dla 16-bitowych IBM-PC, pracujących pod systemem PC-DOS.

Spodziewając się dużego zainteresowania programem, skonsultowaliśmy z autorem tekstu możliwość uzyskiwania przez Czytelników muMATa. Otóż właściciel (zapewne nie jedyny w kraju) bacząc na poniesione koszty nie zamierza go rozdawać. Nie mniej nadesłane pod adresem redakcji propozycje wymiany na inne programy zbliżonej klasy będą wnikliwie rozpatrzone.

Obliczenia symboliczne

Wszyscy przyzwyczailiśmy się już do użycia komputera jako nieco bardziej skomplikowanego liczydła: wszelkie problemy numeryczne (typu wyliczenia wartości całki, rozwiązania układu równań czy znalezienia wartości własnych macierzy) potrafimy rozwiązywać szybko i stosunkowo dokładnie. Stosunkowo, bo wiele problemów, napotykanych przez naukowców i studentów nie da się policzyć d o k ł a d n i e. Typowym przykładem są wszelkie równania, w których występuje liczba **pi** (3.1415... gdzie kropki oznaczają nieskończone rozwinięcie). Oczywiście w praktyce interesuje nas tylko skończona dokładność rozwiązań, bo pomiary na podstawie których opisujemy otaczający nas świat również mają skończoną dokładność. Czasami jednak istotna jest nie tylko przybliżona wartość całki, ale też jej postać analityczna. Żartobliwym przykładem może być egzamin, w którym złośliwy asystent użył funkcji, dającej rozwiązania zależne od **pi**. Nic nam wtedy nie pomoże najlepszy komputer — trzeba mozolnie rozwiązywać problem, myląc się zwykle po drodze wiele razy. W praktyce jedynie analityczna postać funkcji pozwala na podjęcie prób uogólnienia mechanizmów badanego zjawiska.

Od niedawna sytuacja uległa jednak zasadniczej zmianie. Dzięki pracy grupy informatyków z MIT, rozpoczętej w roku 1969, po dziesięciu latach (ponad 100 osobolat pracy) pojawił się program MACSYMA. Potrafił on manipulować s y m b o l i c z n y m i wyrażeniami algebraicznymi, abstrahując od ich wartości numerycznej. Na przykład, MACSYMA potrafi wyznaczyć wyrażenie analityczne będące wynikiem całkowania skomplikowanej funkcji. Niestety, program w obecnej wersji składa się z ponad 300 tysięcy linii kodu w języku LISP i może być uruchomiony tylko na dużych systemach komputerowych (np. VAX firmy DEC). Co gorzej, program jest bardzo drogi: licencja na używanie programu kosztuje ok. 2000 dol. plus liczne opłaty za pomoc techniczną, konsultacje itp.

Dopiero stosunkowo niedawno pojawiły się uproszczone wersje tego programu na komputery osobiste. Pojawiły się one też już w Polsce! Nieliczni jeszcze szczęśliwcy, którzy je posiadają, mogą za przyciśnięciem paru klawiszy uzyskać wyniki jakie dotąd były okupione wielogodzinnymi rachunkami i w dodatku często obarczone błędami.

A oto przykładowe użycie programu muMAT na komputerze IBM PC:

```
INT(X^2+X+1,X);
@: X + X^2/2 + X^3/3
```

Zadaliśmy stosunkowo łatwą funkcję i poprosiliśmy komputer o jej scałkowanie. Po kilku sekundach otrzymaliśmy powyższy wynik. Możemy go sprawdzić zlecając wykonanie odwrotnej operacji:

```
? DIF(@,X,1);
@: 1 + X + X^2
```

Nie przypadkowo jako ilustrację działania programu wybrałem całkowanie symboliczne. Wszyscy, którzy zetknęli się z analizą matematyczną, wiedzą, że różniczkowanie czy rozwiązywanie równań algebraicznych podlega prostym regułom. Nauka metod rozwiązywania tego typu problemów polega w gruncie rzeczy na przyswojeniu sobie tych reguł. Z całkowaniem problem jest nieco bardziej skomplikowany. Wiemy, że różniczkowanie wyniku powinno dać funkcję podcałkową. Jest to jednak jedyna reguła i nie daje ona żadnych wskazówek co do metody postępowania. Okazało się jednak, że dla całek nieoznaczonych, które dadzą się rozwiązać w postaci skończonej (a nie w postaci nieskończonego szeregu), można podać ogólną postać algebraiczną rozwiązania (dla „materializacji" tego przewidywania potrzebne były doświadczenia wielu pokoleń matematyków!). Dalej program poprzez różniczkowanie znajduje odpowiednie współczynniki i podaje wynik jako funkcję. W wielu przypadkach możemy też podać j a k a jest postać szeregu nieskończonego. Jako ciekawostkę należy podać fakt, że eksploatując program stwierdzono, iż większość tablic całek nieoznaczonych zawiera mniej więcej 10 procent błędnych odpowiedzi, a są i takie tablice gdzie co c z w a r t a całka jest źle policzona! A oto przykład całkowania, które daje w wyniku szereg nieskończony (całka z rozkładu Gaussa):

```
? INT(#E^(-X^2),X);
@: #PI^(1/2) ERF (X)/2
```

Jako wynik otrzymujemy dobrze znaną funkcję błędu. Proszę zwrócić uwagę, że program podał stałą jako **pi** a nie jako liczbę! A oto kilka innych przykładów całkowania:

```
? INT(X^3/(1+X),X);
@: 11/6 + X - X^2/2 + X^3/3 - LN (1 + X)

?INT((LN(1. /(1+X))^2),X);
@: 2 + 2 X + X LN (1 + X)^2 - 2 X LN (1 + X) + LN (1 + X)^2 - 2
LN (1 + X)
```

Prawda, że te wyniki robią wrażenie?!
Niestety, ze względu na niewielką pojemność pamięci komputerów osobistych (w tym przypadku trzeba i tak użyć 256 KB pamięci), nie wszystkie całki dadzą się policzyć:

```
? INT(TAN(LN(1/(1+X))),X);
@: -INT (SIN LN (1 + X)/COS LN (1 + X), X)

? INT(SIN(LN(1+X)),X);
@: INT (SIN LN (1 + X), X)

? INT(LN SIN(1+X),X);
@: INT (LN SIN (1 + X), X)
```

Nie wiem, czy takie całki przekraczają możliwości IBM PC, czy też może po prostu mają jedynie rozwiązania w postaci szeregów nieskończonych — program nie daje żadnej wskazówki, a szkoda...

Program muMAT potrafi wiele — manipulowanie macierzami, rozwiązywanie równań różniczkowych (oczywiście poprzez całkowanie symboliczne!), rozwiązywanie równań

algebraicznych — to typowe przykłady problemów nie przysparzających kłopotów programowi muMAT.

Reklamy programów symbolicznego manipulowania zmiennymi podają przykład, który zapiera dech w piersiach. Otóż w XIX wieku francuski astronom Charles Delaunay przez 10 lat wyznaczał analityczny wzór na ruch Księżyca wokół Ziemi. Problem nie jest prosty (ruch trzech ciał w polu grawitacyjnym) i dlatego następne dziesięć lat trwało sprawdzanie wyników obliczeń. Prace zostały ukończone w 1867 roku. Od tego czasu tablice, oparte na wynikach obliczeń Delaunaya, były używane przez pokolenia astronomów. Dopiero w roku 1970, po 20 godzinach pracy program MACSYMA znalazł błąd! Całe szczęście, że dotyczył on tylko czynnika przed jednym z wyrażeń (cały wzór zajmuje wiele stron) zamiast 23/16 powinno być 33/16 i ten niewielki błąd nie powoduje większych różnic w końcowych wynikach. A więc proporcja: 20 lat i 20 godzin! Czym się to wszystko skończy? Może już niedługo będziemy świadkami masowego poprawiania starych prac naukowych?! W każdym razie fizyka teoretyczna i wiele innych specjalności, szczególnie w matematyce stosowanej, gdzie dotąd królowały kartka papieru i ołówek, zyskały potężne narzędzie pracy. Miejmy nadzieję, że i w Polsce komputery profesjonalne będą dysponowały programami symbolicznego obliczania wyrażeń.

JAKUB TATARKIEWICZ

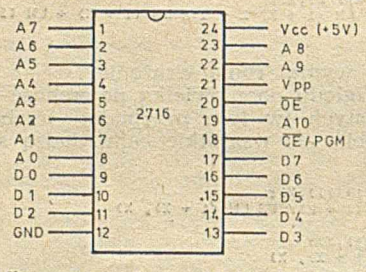
Opublikowany w mikroKLANIE 12 schemat programatora pamięci 2716 wywołał falę listów i telefonów z prośbami o dane katalogowe zarówno samego układu EPROM, jak i algorytmu programowania. Przykro odmawiać potrzebującym, ale redaktor mikroKLANU nie jest w stanie odwalić roboty, którą powinny wykonywać branżowe ośrodki informacji naukowej i technicznej. Zresztą jeśli je o to poprosić, to ją wykonują (np. BOINTE, Warszawa, ul. Marynarska 10), tyle że za kwoty mocno nadwyrężające prywatną kieszeń, a na dodatek raczej niespiesznie.

Ponieważ układy EPROM są dość powszechnie stosowane w systemach mikroprocesorowych, a układ 2716 należy do tych najpopularniejszych, sądzimy, że zamieszczony opis rozwiąże problemy wielu Czytelników — tym bardziej, że został wzbogacony o informacje, których nie ma w katalogu.

W przyszłości nadal będziemy zamieszczać opisy tych układów, o które najczęściej dopytują się Czytelnicy (lub układów, które naszym zdaniem należy lansować), ale prosimy pamiętać, że mikroKLAN nie jest drukowanym w odcinkach katalogiem.

Pamięć EPROM 2716

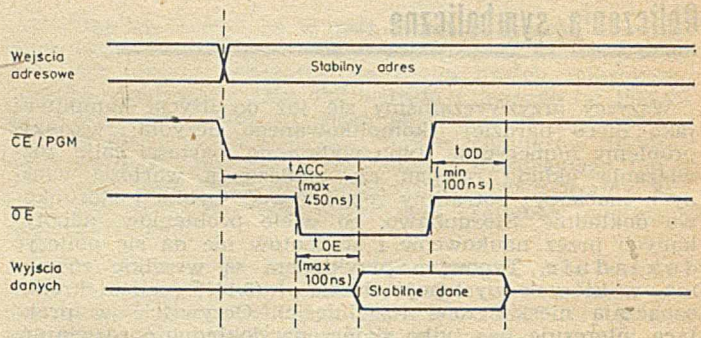
Topografia wyprowadzeń układu 2716 przedstawiona została na rysunku 1. Układ ma organizację 2 K × 8 bitów i jest następcą układów 2704 i 2708 firmy INTEL. Od poprzedników odróżnia go nie tylko większa pojemność, ale też pojedyncze napięcie zasilania (+5V) i uproszczony algorytm programowania. Sposób programowania jest identyczny jak dla późniejszych układów EPROM 2732, 2764, 27128 i wreszcie 27256.



Rys. 1. Topografia wyprowadzeń układu 2716 firmy INTEL

Realizacja cyklu odczytu zawartości określonej komórki pamięci wymaga wymuszenia niskiego poziomu na wejściu CE/-PGM. Podanie poziomu wysokiego na to wejście przełącza układ w tryb „uśpienia" (ang. power down). W tym trybie pobór mocy redukowany jest o ok. 75%, co poważnie odciąża zasilacz, ale też powoduje silne impulsy prądowe przy przechodzeniu układu w stan aktywny — może być to źródłem zakłóceń dla innych układów.

Dla realizacji odczytu niezbędne jest również wymuszenie poziomu niskiego na wejściu OE/ (ang. Output Enable). Wejście to aktywizuje wyjściowe bufory danych. Jak wynika z wykresu czasowego (rys. 2), dane mogą być odczytywane po ok. 450 ns (wersja standardowa) do ustalenia się informacji na wejściach adresowych. Wykorzystywanie trybu uśpienia w czasie, gdy mikroprocesor nie realizuje odczytu z układu, nie wnosi więc dodatkowych opóźnień czasowych. Wejścia CE/-PGM i OE/ mogą być sterowane tym samym sygnałem. W praktyce na wejście CE/-PGM podawany jest sygnał z dekodera adresów, natomiast wejście OE/ sterowane jest strobem odczytu z pa-



Rys. 2. Przebiegi czasowe przy realizacji cyklu odczytu pamięci

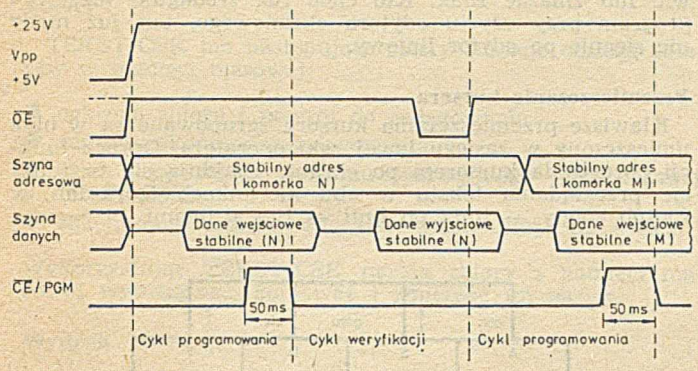
mięci. Ponieważ dla adresów zajmowanych przez pamięć EPROM nie są wykonywane cykle zapisu, nie powoduje to niepotrzebnego pobudzania układu. „Fałszywe" pobudzenie może jednak wystąpić przy zbieżności adresu generowanego przy operacji we-wy z adresem zajmowanym przez EPROM.

Należy zwrócić uwagę, że prawidłowa praca układu wymaga napięcia +5V na wejściu Vpp (nóżka 21). Podanie na to wejście stanu logicznego „1" zazwyczaj okazuje się niewystarczające. Stanowi to pewną niedogodność przy projektowaniu konstrukcji, w których w miejsce układu 2716 można by ewentualnie stosować 2732 (na nóżkę 21 podawany jest wtedy A11).

Przy realizacji odczytu, na nóżce 21 może występować napięcie wyższe od +5V (do +25V) jednak na wejściu CE/-PGM musi być wtedy (w sposób ciągły) wymuszony poziom niski.

Programowanie układu wymaga podania na wejście Vpp napięcia +25V. Każda z komórek układu może być programowana niezależnie od pozostałych (komórka zawiera 8 bitów). W tym celu należy na szynie adresowej wymusić adres żądanej komórki, a na szynie danych informację, która ma być wpisana. Aby dana została zapamiętana na wejściu CE/-PGM należy przez ok. 50 ms wymusić poziom wysoki. Weryfikację zapisu można zrealizować wymuszając poziom niski na wejściu OE/ (rys. 3). W tym czasie

na wejściu Vpp może być utrzymane napięcie +25V, pod warunkiem jednak, że na CE/-PGM będzie stan logiczny „0".

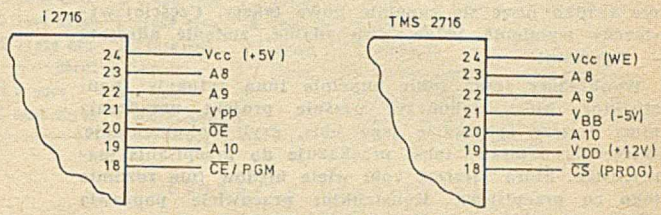


Rys. 3. Przebiegi czasowe przy programowaniu i weryfikacji zawartości pamięci

Układ 2716 po wymazaniu zawartości promieniowaniem ultrafioletowym (lub fabrycznie nowy) ma wszystkie bity ustawione w stan logiczny „1". Programowanie polega na wymuszeniu przejścia żądanych bitów w stan logiczny „0". Jeżeli w trakcie programowania niektóre bity pozostawiono w stanie logicznym „1", to można je w dowolnym momencie „doprogramować" postępując analogicznie jak przy pierwszym programowaniu, z tym, że oczywiście informacja wejściowa musi mieć „0" na odpowiednim bicie. Stanu logicznego bitów, które zostały zaprogramowane jako „0" nie można już zmienić w inny sposób, niż wymazanie zawartości całego układu promieniowaniem ultrafioletowym.

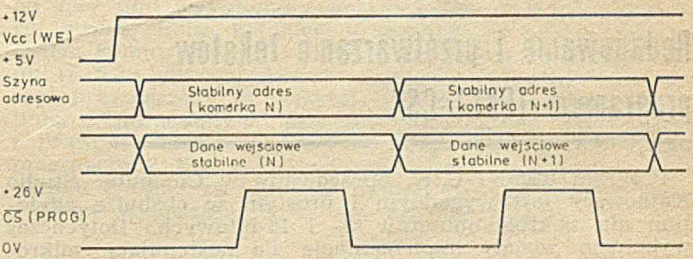
Programowanie i wymazywanie zawartości układów EPROM może być wykonywane wielokrotnie, jednak po każdej takiej operacji pozostaje ślad w postaci pewnej ilości rozproszonego ładunku. Zbyt duża ilość ładunku rozproszonego zmniejsza skuteczność programowania i w efekcie końcowym powoduje „zużycie" układu. Standardowa procedura programowania przewiduje, że impuls na wejściu CE/-PGM trwa ok. 50 ms. Powoduje to, że czas programowania całego układu zbliża się do dwóch minut. Przy układach o większej pojemności czas ten wydłuża się proporcjonalnie sięgając ok. pół godziny dla 27256. Skłoniło to producenta do poszukiwania algorytmów o krótszym czasie programowania. Algorytm taki opisany zostanie w jednym z następnych wydań mikroKLANU.

Układ EPROM 2716 produkowany jest obecnie przez wiele różnych firm i z reguły w oznaczeniach zawarte są cyfry 2716. Warto jednak zwrócić uwagę, że firma TEXAS INSTRUMENTS użyła oznaczenia TMS 2716 dla pamięci EPROM, która zasadniczo różni się od układu firmy INTEL. Podstawową różnicą jest konieczność stosowania trzech napięć zasilających dla układu TMS 2716 (rys. 4).



Rys. 4. Różnice w topografii wyprowadzeń układów i2716 (firma INTEL) oraz TMS 2716 (firma TEXAS INSTRUMENTS)

Całkowicie odmienny jest też algorytm programowania. Poszczególne komórki muszą być programowane w sposób sekwencyjny impulsami o czasie trwania od 0,1 do 1 ms. Po przejściu wszystkich komórek procedura jest powtarzana aż do momentu, gdy sumaryczny czas trwania impulsów programujących będzie dla każdej komórki dłuższy od 100 ms (rys. 5). Algorytm programowania układu TMS 2716 jest więc bardzo zbliżony do algorytmu wykorzystywanego przy programowaniu układów 2708 firmy INTEL. Zbliżona topografia wyprowadzeń układów 2708 i TMS 2716 stwarza możliwość wykorzystania tego drugiego w konstrukcjach, gdzie planowane było wykorzystywanie pamięci EPROM 2708. Należy w tym miejscu podkreślić, że włożenie standardowego układu 2716 w miejsce 2708 prowadzi do jego uszkodzenia.



Rys. 5. Przebiegi czasowe przy programowaniu układu TMS 2716

Firma TEXAS INSTRUMENTS produkuje też układ pamięci EPROM zbliżony do INTELowskiego 2716. Oznaczany jest on jednak TMS 2516. W mikroKLANIE 12 podano różnicę w topografii wyprowadzeń układów 2716 i TMS 2516.

AJP

Po sukcesie komputera CPC 464 firma AMSTRAD wprowadziła nowy model CPC 664. Zamieniono w nim magnetofon kasetowy na napęd dyskietek 3". Wkrótce jednak rozeszły się wieści o nowszym modelu z rozszerzoną pamięcią RAM. Zmiana ta poważnie zwiększyła wartość użytkową komputera: bardziej popularne programy, pracujące pod nadzorem systemu CP/M, wymagają nieco większego obszaru pamięci operacyjnej niż mogą to zapewnić modele 464 i 664. Powoduje to konieczność podziału programu na segmenty sprowadzane — stosownie do sytuacji — z dysku. W rezultacie program pracuje znacznie wolniej.

Aby nie hamować sprzedaży modelu 664 ogłoszono, że nowa wersja będzie początkowo oferowana tylko na rynek amerykański. Mimo tych zapewnień, w sierpniu na rynek brytyjski trafił CPC 6128 i to w cenie modelu 664. Stało się więc jasne, że model CPC 664 zostanie wycofany z produkcji. Wywołało to wiele protestów ze strony nabywców (producenci oprogramowania przerwali prace nad adaptacjami). Narzekali też sprzedawcy, którym w magazynach zostały modele 664. Jednak duży popyt na CPC 6128 szybko wyrównał im straty.

Pod koniec września 1985 r. AMSTRAD znowu wszystkich zaskoczył: wprowadzono kolejny nowy model — PCW JOYCE 8256. O ile poprzednie wersje uwzględniały fakt, że większość komputerów domowych kupuje się dla zabawy, to z charakterystyki PCW 8256 wynikało, że jest on przeznaczony dla tych, którzy traktują sprawę poważnie. Zielony monitor (choć z większą rozdzielczością) studzi niewątpliwie zapędy fanatyków gier. Dysk o większej pojemności, pamięć RAM 256 KB i drukarka w ramach konfiguracji podstawowej to sprzęt godny zastosowań profesjonalnych. Swoją propozycję AMSTRAD opiera na przeświadczeniu, że odbiorcy wyrośli już z gier i szukają komputera-pomocnika. Przetwarzanie tekstów i baza danych to aplikacja przydatna dla każdego.

Wygląda na to, że AMSTRAD zapoczątkuje rewolucję podobną do tej, jaką kiedyś zrobił ZX 80, tyle że o innym ciężarze gatunkowym. Zabawki wezmą się do roboty.

Redagowanie tekstów nie jest wyłącznie domeną redaktorów. Większa część tzw. prac biurowych sprowadza się do przetwarzania tekstów. Nie chodzi tu tylko o ułatwienie poprawiania błędów, ale często też o wykorzystywanie całych fragmentów napisanych już przy innej okazji. Biurowa praktyka wskazuje, że niezmiernie rzadko pisze się zupełnie nowe teksty. Częściej wystarczy wymienić jedno, dwa zdania, zmienić adresata — i gotowe!

Wyobraźmy teraz sobie zupełnie inną sytuację: konstruktor, który zakończył właśnie projekt urządzenia musi jeszcze sporządzić jego opis, czyli dokumentację. Odręcznie napisany tekst przekazuje do przepisania maszynistce, która niestety robi wiele błędów (nie rozumie tego co przepisuje). Konstruktor pracowicie poprawia błędy maszynopisu i przekazuje tekst do ponownego przepisania. Gdy tekst wraca, okazuje się, że choć stare błędy zostały poprawione to pojawiły się nowe. „Zabawa" może powtarzać się wielokrotnie. Gdyby maszynistka wprowadziła tekst do komputera, to bez większych kłopotów konstruktor poprawiłby ewentualne błędy (nie ma potrzeby powtórnego przepisywania) i materiał byłby gotów do rozpowszechniania.

Niedawno opisywaliśmy procesor tekstów dla ZX SPECTRUM. W poniższym tekście prezentujemy program nadający się już do zastosowań profesjonalnych — na profesjonalnych komputerach. Podstawową wadą programu jest brak możliwości stosowania typowo polskich znaków (np. ą, ę, ł itp.). Taka jest niestety cena przenoszenia programu między różnymi komputerami. Oczekujemy jednak, że w następnej generacji programu CSK udoskonali go przynajmniej dla kilku typów mikrokomputerów lub może nawet przygotuje specjalny moduł adaptacyjny, pozwalający użytkownikowi przystosować program do specyfiki posiadanego komputera i drukarki. Czekamy niecierpliwie!

Redagowanie i przetwarzanie tekstów programem TEKST-CSK

Program TEKST-CSK, opracowany w Computer Studio Kajkowscy jest wygodnym i prostym w obsłudze edytorem dla mikrokomputerów 8- i 16-bitowych. Dotychczas wykonane zostały implementacje na następujące mikrokomputery: APPLE II, LIDIA, ELWRO-513/523, ROBOTRON-5120/5130, IMP-85, MK-4, RTDS-8 i MC oraz IBM PC, Olivetti M24, LIDIA 2, EMIX, IMP-86 i inne kompatybilne z APPLE II lub IBM PC. Oprogramowanie może działać na sprzęcie spełniającym następujące wymagania:

— mikroprocesor Z80, 8080, 8085, 8086 lub 8088
— minimum 48 KB pamięci RAM
— pamięć masowa na dyskach elastycznych
— alfanumeryczny monitor ekranowy o szerokości 64 lub 80 znaków w linii
— drukarka.

TEKST-CSK jest programem bardzo przydatnym do prac biurowych takich, jak: prowadzenie korespondencji, przygotowywanie pism wewnątrz zakładowych, tworzenie i przechowywanie sprawozdań oraz innych dokumentów, redagowanie ulotek reklamowych itp. Jako przykład zastosowania TEKST-CSK można podać niniejszy artykuł zredagowany i wydrukowany [1]) (w maszynopisie) przez autora przy użyciu opisywanego programu na mikrokomputerze IBM PC.

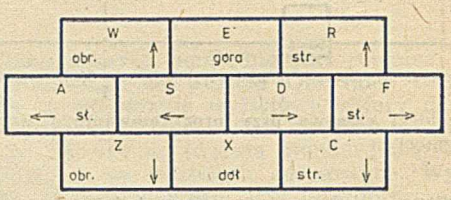
TEKST-CSK służy również do redagowania programów źródłowych w dowolnym języku programowania. W przypadku, gdy mikrokomputer stanowi inteligentną końcówkę dużego komputera, ekranowy program redakcyjny staje się nieodzowny.

[1]) Niestety INFORMATYKA nie dysponuje tzw. składem komputerowym i artykuł został przygotowany do druku tradycyjnymi metodami znanymi z czasów Gutenberga (przypis AJP)

Pojęcie „ekranowy" oznacza, że czynność redagowania (dopisywanie, wymazywanie, poprawianie) odbywa się w dowolnym miejscu ekranu, na którym wyświetlony jest wybrany fragment tekstu. Zatem ekran jest jak kartka papieru, na której można w każdym miejscu dopisać, poprawić lub zmazać znak. Kto choć raz zredaguje tekst lub program przy użyciu edytora ekranowego, ten już nigdy nie sięgnie po edytor liniowy.

Przemieszczanie kursora

Klawisze przemieszczania kursora zgrupowane są w blok umieszczony w zasięgu lewej ręki operatora. Oprócz funkcji poruszania kursorem po ekranie znajdują się tu funkcje przeglądania tekstu w obu kierunkach (góra-dół), w sposób ciągły — linia po linii — lub stronami.



Program TEKST-CSK ma 10 rejestrów, w których można zapamiętać dowolne adresy odpowiadające różnym pozycjom kursora. Można je wykorzystać w dowolnym momencie. W każdej chwili można także wykonać skok do jednego z pięciu ważnych adresów: na początek lub na koniec zbioru, na początek lub na koniec bloku i do poprzedniej pozycji kursora (powrót).

Redagowanie

Pozycja kursora jest istotna z tego względu, że wszelkie czynności korekcyjne odbywają się właśnie w tym miejscu. Naciskając odpowiedni klawisz można:

— zmazać: znak z lewej lub z prawej strony, słowo, część linii od kursora do końca, część linii od początku do kursora, całą linię
— dopisać: znak, słowo, linię
— poprawić dowolny znak w tekście.

Przewidziano również funkcję automatycznego wyszukiwania i wymiany dowolnej frazy na inną.

Formatowanie

Ekran dla TEKST-CSK ma szerokość 80 lub 64 znaków. Teksty redagowane przy użyciu programu mogą mieć dowolną szerokość. Deklaracja szerokości tekstu oraz jego rozmieszczenie na monitorze odbywa się poprzez ustawienie lewego i prawego marginesu. Uwzględnianie marginesów polega na przeniesieniu wyrazu nie mieszczącego się przed prawym marginesem i przesunięciu pozostałych wyrazów tak, by pole pomiędzy marginesami było równomiernie wykorzystane. Do formatowania według marginesów należy zaliczyć także funkcję centrowania, która ustawia tekst w środku pola między marginesami (istotne przy tytułach).

Użytkownik programu TEKST-CSK ma do dyspozycji różne typy tabulatorów. Tabulator stały (o długości skoku — 8) dostępny jest podczas redagowania zarówno tekstów jak i programów źródłowych. Tabulatory zmienne (ustawiane przez użytkownika) można stosować jedynie przy redagowaniu tekstów. Rozróżniane są dwa rodzaje tabulatorów zmiennych: dla słów oraz dla liczb.

Bloki

Blok jest to część zbioru wyznaczona dwoma adresami (pozycja kursora) — początkowym i końcowym. Adres początkowy musi oczywiście poprzedzać adres końcowy. Zaznaczony wcześniej blok można wymazać, przenieść w in-

ne miejsce zbioru, przekopiować w miejsce ustawienia kursora, zapisać w oddzielnym zbiorze pamięci masowej lub odczytać ze zbioru na dysku.

Zbiory

TEKST-CSK ma następujące cztery funkcje obsługi zbiorów w pamięci masowej:

- ochrona zbioru i powrót do redagowania
- zapis i zakończenie redagowania
- zapis i powrót do systemu operacyjnego (koniec pracy z TEKST-CSK)
- zakończenie redagowania bez zapisu zbioru na dysku.

Wykorzystując TEKST-CSK można zbiory z pamięci masowej wymazywać, kopiować i zmieniać im nazwy.

Wydruk

Końcowym efektem redagowania jest sporządzenie trwałej kopii tekstu na drukarce. W trakcie wydruku realizowane są dyrektywy (linie tekstu, które zaczynają się od kropki). Formatowanie wydruku zależne jest od typu przyłączonej do mikrokomputera drukarki i polega na dzieleniu tekstu na strony z uwzględnieniem: długości stron, marginesów, numeracji, komentarzy nad i pod tekstem, szerokości i wysokości liter, odstępu między liniami, koloru taśmy, zestawu znaków drukarskich itp.

Język przetwarzania tekstowego

Program TEKST-CSK umożliwia również wykonywanie bardziej skomplikowanych czynności przetwarzania tekstowego. W przypadku, gdy zachodzi potrzeba wielokrotnego wydruku tego samego zbioru, podobnych tekstów, łączenia zbiorów w trakcie drukowania lub skorzystania ze zgromadzonych danych — można zastosować język przetwarzania tekstowego. W ten sposób zautomatyzowano edycję pism o podobnej treści i formie, a różniących się tylko fragmentami.

* * *

W artykule opisano tylko najważniejsze z listy ok. 150 funkcji redakcyjnych. Ich wywołanie odbywa się poprzez naciśnięcie jednego lub dwóch klawiszy. Funkcje podzielone są na sześć grup według przeznaczenia. Dla każdej grupy na ekranie wyświetlane jest tzw. menu, czyli lista funkcji z opisem. Po nabraniu wprawy w operowaniu programem, można zrezygnować z wyświetlania komentarzy pomocniczych. Na rysunku 2 przedstawiono kopię obrazu monitora w trakcie działania TEKST-CSK. W pierwszej od góry linii wyświetlany jest aktualny stan (nazwa zbioru, strona, linia, kolumna) i komunikaty dla operatora. Osiem kolejnych linii, to obszar dla jednego z sześciu menu (z tego obszaru można zrezygnować, rozszerzając obszar przeznaczony dla tekstu). Pod menu narysowany jest linial z zaznaczonymi pozycjami tabulatorów oraz lewego i prawego marignesu. Pod linialem znajduje się obszar przeznaczony dla redagowanego tekstu. Na samym dole ekranu opisane są klawisze funkcji specjalnych dla komputerów wyposażonych w tego rodzaju funkcje.

```
A:APEL.ART  STR. 1 LIN. 1 KOL 01                DOPIS. WL
          ( ( (  G L O W N E   M E N U  ) ) )
------ Przemieszczenia ------¦- Kasowanie -¦---- Pomocnicze ---¦--- Inne menu --
^S lewy znak  ^D prawy znak  ¦^G prawy zn.¦^I tab    ^B format ¦^J dla objasnien
^A lewe slowo ^F prawe slowo ¦^L lewy zn.¦^V dopisywanie       ¦^K dla blokow
^E linia nad  ^X linia pod   ¦^T slowo   ¦^L szukanie         ¦^U dla skokow
------- Obrot ekranu --------¦^Y linii   ¦^R koniec akapitu   ¦^P dla wydruku
^Z w gore     ^W w dol       ¦           ¦^N posta linia      ¦^O dla ekranu
^C nast.obraz ^R poprz.obraz ¦           ¦^U zatrzymanie      ¦
L----¦----¦----¦----¦----¦----¦----¦----¦----¦----¦-------R
```

```
                    S z a n o w n y   C z y t e l n i k u

                    Autor artykułu prosi o kierowanie

                    swoich uwag i pytań na adres :

                                    Antoni Urban

                                    ul. Balladyny 3b

                                    81-524 Gdynia - Orłowo
```

10BJAS. 2AKAPIT 3L.MARG 4P.MARG 5PODKR. 6CIEN. 7POC.BL 8KON.BL 9POC.ZB 10KON.ZB

ANTONI URBAN
CSK-Gdynia

Realizując proste systemy mikroprocesorowe przeznaczone do celów sterowania stajemy często przed koniecznością dołączania wielu układów, które wykorzystywane są w ograniczonym zakresie. Przykładowo, potrzeby w zakresie pamięci typu RAM sprowadzają się zazwyczaj do kilkudziesięciu bajtów. Układy z rodziny MCS 85 projektowane były często właśnie z myślą o tego typu zastosowaniach.

Połączenie kilku funkcji w jednym układzie (np. pamięć RAM, port we-wy, zegar) nie tylko upraszcza konstrukcję, ale też przyczynia się do zwiększenia niezawodności. Mniejszy pobór prądu zasilającego może uprościć konstrukcję zasilacza, a zaoszczędzone miejsce — spowodować zmniejszenie gabarytów urządzenia.

W prezentowanym rozwiązaniu Jerzy Orkiszewski proponuje umieszczenie układu w przestrzeni adresowej we-wy. Zastępując linie I-OR/ i I-OW/ przez odpowiednio MEMR/ i MEMW/, i ewentualnie rozbudowując dekoder, możemy umieścić układ w przestrzeni adresowej pamięci, co pozwoli wykorzystywać w programach znacznie szerszy zestaw rozkazów. Jednak nawet taka zmiana nie pozwoli wyeliminować stosunkowo poważnej wady rozwiązania: zawarta w układzie 8155 pamięć RAM nie może być wykorzystana jako stos. Jeżeli więc chcemy korzystać z przerwań lub podprogramów musimy niestety dołączyć dodatkową pamięć RAM.

Układy z multipleksowaną szyną danych w systemie 8080

Magistrale systemowe mikroprocesorów podzielić można na dwie grupy. Pierwsza obejmie rozdzielone szyny adresów i danych czyli takie, w których dane i adresy przesyłane są po osobnych liniach. Przedstawicielami tej grupy są popularne mikroprocesory, takie jak: 8080, Z80, 6800 czy 16-bitowy 68000. W drugiej grupie dla szyny adresowej i danych wykorzystywane są te same linie. Szyna danych służy tu do przesyłania części adresu a następnie słowa danych. Odpowiednie impulsy strobujące pozwalają rozróżnić typ aktualnie podawanej informacji. Tego typu szyny określane są mianem „multipleksowanych". Multipleksowane szyny występują m.in. w 8085, układach z serii 6805, 8086 i większości mikroprocesorów zrealizowanych w technologii CMOS.

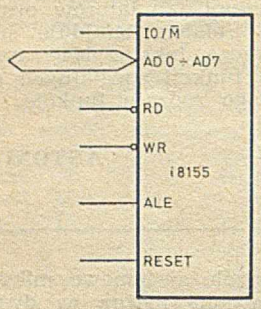
Każdej z tych grup towarzyszy rodzina układów peryferyjnych, zazwyczaj przystosowanych do multipleksowanej szyny danych i adresów. Często są to układy zrealizowane jako

CMOS i wykazują znane zalety tej technologii, Szczególnie interesujące są układy peryferyjne CMOS z multipleksowaną szyną danych, oferowane przez firmę MOTOROLA. Są one zaopatrzone w specjalny interfejs nazwany MOTEL (od nazw MOTorola i IntEL). Pozwala to na prostą realizację współpracy z szyną mikroprocesora typu 8085 jak i 6805. To, który z mikroprocesorów steruje układem rozpoznawane jest przez interfejs automatycznie, na podstawie zależności czasowych na szynie.

Dołączenie „zwykłego" układu peryferyjnego do mikroprocesora z multipleksowaną szyną danych nie nastręcza zwykle większych problemów. Wystarczy rozdzielić obie szyny za pomocą rejestru zatrzaskowego. Znacznie więcej problemów sprawia sytuacja odwrotna, np. połączenie mikroprocesora typu 8080 z układem multipleksowanym z rodziny 8085.

Opisany dalej przykład rozwiązania tego problemu dotyczy połączenia procesora 8080 z układem 8155. Podana metoda pozwala łączyć w zasadzie dowolny, „zwykły" procesor z każdym układem we-wy lub pamięci z multipleksowaną szyną danych i adresów.

Układ 8155 jest elementem z rodziny MCS-85. Kostka zawiera 256 bajtów pamięci RAM, dwa 8-bitowe i jeden 6-bitowy port we-wy oraz licznik. Zestaw linii łączących układ z systemem przedstawia rysunek 1.



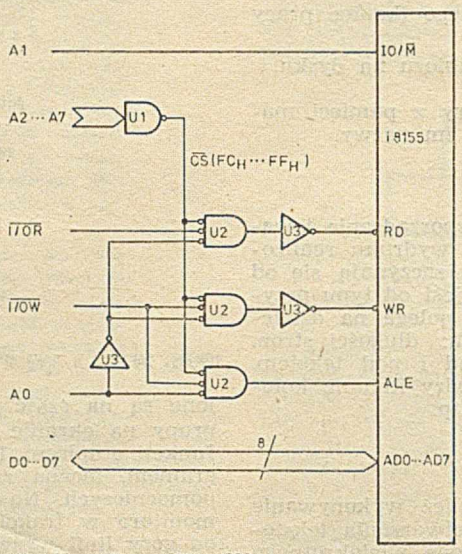
Rys. 1. Linie interfejsu systemowego dla układu 8155

Adres przesyłany w pierwszej części cyklu dostępu zatrzaskiwany jest sygnałem ALE. W drugiej części cyklu, dana przesyłana jest do lub z układu. O kierunku przesłania decydują stany logiczne na wejściach RD/ i WR/. O tym, czy adres zatrzaśnięty sygnałem ALE dotyczył wewnętrznej pamięci RAM czy też wybierał któryś z rejestrów we-wy albo licznika decyduje stan podany na linię IO-M/.

Schemat dołączenia układu 8155 do szyny mikroprocesora 8080 przedstawia rysunek 2. Mikroprocesor „widzi" układ jako zestaw czterech komórek przestrzeni adresowej we-wy. Dwie spośród tych komórek to zatrzaski adresowe. Wpis do tych rejestrów odbywa się poprzez wygenerowanie przez układy logiki adaptacyjnej impulsu ALE. Aby sygnał taki powstał muszą być spełnione następujące warunki:

a) procesor adresuje układ 8155, czyli aktywny jest dekoder, do którego dołączone są linie A2...A7 oraz aktywny jest sygnał IOW/

b) na linii A0 wymuszony jest stan logiczny 0.



Rys. 2. Przykładowy sposób podłączenia układu 8155 do szyny systemowej mikroprocesora 8080

Stan linii A1 decyduje czy wysłany bajt danych (będący adresem wewnętrznym dla układu 8155) stanowi odwołanie do jego części pamięciowej czy też we-wy.

Pozostałe dwie komórki to pseudorejestry danych. Wpis bądź odczyt danych z tych komórek odbywa się poprzez wygenerowanie odpowiednich impulsów na wejściach WR/ lub RD/ układu 8155. Sposób generowania tych sygnałów jest podobny jak dla ALE, z tym, że na linię A0 musi zostać podany poziom wysoki. W przykładzie przedstawionym na rysunku 2, poszczególnym komórkom przyporządkowane są następujące adresy:

— zatrzask adresowy dla pamięci FC_H (A7...A2=1, A1=0, A0=0)

— pseudorejestr danych pamięci FD_H (A7...A2=1, A1=0, A0=1)

— zatrzask adresowy dla we-wy FE_H (A7...A2=1, A1=1, A0=0)

— pseudorejestr danych we-wy FF_H (A7...A2=1, A1=1, A0=1)

Współpraca z układem odbywa się według następującego schematu:

a) wysłanie adresu do zatrzasku adresowego

b) odczyt lub zapis danej z (lub) do odpowiedniego pseudorejestru.

Przykładowa procedura odczytu danej z wewnętrznej pamięci układu 8155 (spod adresu 38H) ma następującą postać:

```
MVI  A, 38H  ; adres w akumulatorze
OUT  FCH     ; zapis do zatrzasku
               adresowego
IN   FDH     ; dana do akumulatora
```

Zapisując daną do pamięci możemy wykorzystać program:

```
MVI  A,38H     ; adres w akumulatorze
OUT  FCH       ; zapis do zatrzasku
                 adresowego
MVI  A, DANA   ; dana do akumulatora
OUT  FDH       ; zapis do pseudorejestru
```

Opisana metoda polega na dołączeniu prostego układu kombinacyjnego i nieznacznej rozbudowie programu. W zamian otrzymujemy możliwość wykorzystania całej gamy nowych układów, zazwyczaj wyjątkowo uniwersalnych, co szczególnie w małych systemach jest nie bez znaczenia.

JERZY ORKISZEWSKI
Warszawa

- Wygląda na to, że obietnice firmy ATARI, dotyczące nowej rodziny mikrokomputerów ST, zostały dotrzymane. Model 260 ST „Little Saint" zawiera wbudowany napęd dysków 3,5" i razem z myszą (ale bez monitora) ma kosztować 500 funtów. Na wystawie promocyjnej pokazywano też wersję z 1 MB RAM (!), zapowiadając, że będzie też i wersja z 4 MB RAM. Dla serii ST oferowanych jest już obecnie 80 pakietów programowych. Równocześnie drastycznie obniżono ceny na starsze modele. 800XL z pamięcią 64 KB, drążkiem sterowniczym i modułem pamięci z grami kosztuje tylko 70 funtów. 130XE (128 KB RAM) kosztuje 140, a razem z napędem dysków — 265 funtów. Za 350 funtów można nabyć komplet z drukarką i oprogramowaniem redaktora tekstu.
- Firma SINCLAIR uchroniła się przed bankructwem i wykupieniem przez Roberta Maxwella. Projektowana jest nowa wersja ZX SPECTRUM+ z nowym rozwiązaniem ROM zawierającym ulepszony BASIC, z 128 KB RAM, generatorem dźwięków AY 33910 i o większych możliwościach graficznych. Cenę QL obniżono do 200 funtów.

Ogłoszenia • Ogłoszenia

Oprogramowanie do COMMODORE C-64 piszemy na zlecenie instytucji, osób prywatnych. 27-400 Ostrowiec, skrytka 40
EO/1103/K/85

W poprzednim wydaniu mikroKLANU opublikowaliśmy schemat ideowy niewielkiego systemu zbudowanego na bazie mikroprocesora MOTOROLA 6800. Ten popularny na świecie mikroprocesor jest w naszym kraju stosunkowo mało znany. W tej sytuacji pierwsze kroki z systemem mogą być stosunkowo trudne. Ponieważ mikrokomputer tak naprawdę nabiera życia dopiero z chwilą, gdy jest do niego jakieś oprogramowanie, publikujemy teraz program, który na początku przygód z 6800 może okazać się bardzo przydatny.

Program MONITOR dla mikroprocesora MOTOROLA 6800

Program MONITOR [1]) zajmuje w pamięci system obszar o długości 1KB (od $FC00_H$ do $FFFF_H$). Umożliwia to pobranie przez system wektorów restartu i przerwań bezpośrednio z programu. Zmienne systemowe, tablice adresów i stan rejestrów procesora pamiętane są w obszarze roboczym pamięci RAM (minimum 128 bajtów). Powinien być on umieszczony od adresu $E000_H$. Procedury we-wy obsługują układ **ACIA** — MC6850 umieszczony pod adresami $8008/8009_H$. Program nie wykorzystuje przerwań sprzętowych; są one w pełni dostępne dla użytkownika. Jedyne ograniczenia dotyczą użycia przerwania programowego **SWI**, które wykorzystywane jest przy pracy krokowej. Istnieje jednak możliwość zmiany adresu obsługi tego przerwania, gdyż jest on każdorazowo pobierany z pamięci RAM.

Opis działania programu

Po inicjacji, procesor rozpoczyna działanie od wpisania do licznika rozkazów zawartości komórek o adresach $FFFE/FFFF_H$. Znajduje się w nich adres tzw. „zimnego" startu programu — $FC20_H$. Na początku program przygotowuje w pamięci RAM „notatnik". Przepisywana jest do niego tabelka kodów modyfikowanej części programu, a pozostała część jest zerowana. Następnie programowany jest układ 6850. Słowo programujące (19_H przechowywane pod adresem $FC3C_H$) powoduje: wewnętrzny podział TxC i RxC przez 16; długość transmitowanego słowa 8 bitów + bit stopu + bit parzystości; wyjście RTS/ ustawiane w stan 0; przerwania z nadajnika i odbiornika są zablokowane. Słowo programujące po „zimnym" starcie pamiętane jest w pamięci RAM pod adresem $E019_H$. W tzw. „gorącym" wejściu programu (adres $FC40_H$) wpisywane jest ono do rejestru sterującego ACIA. Wskaźnik stosu ustawiany jest na adres $E078_H$.

Następnie program wysyła do terminala zgłoszenie gotowości (znak: *) i przechodzi do oczekiwania na dyrektywę. Po jej odebraniu przegląda tablicę adresów i kodów dyrektyw. Tablica ta (zajmująca obszar: $FFD1_H$-$FFF7_H$) zorganizowana jest następująco: bajt kodu ASCII dyrektywy + dwa bajty adresu procedury obsługi. Gdy po przejrzeniu całej tablicy dyrektywa nie zostanie rozpoznana, program wraca do pętli oczekiwania.

System dopuszcza wprowadzenie ośmiu różnych adresów pułapek. Dojście wykonywanego programu do adresu pułapki powoduje wykonanie podprogramu obsługi przerwania programowego. Podprogram ten drukuje zawartość rejestrów procesora w chwili napotkania pułapki. Po przerwaniu znajdują się one na stosie. Wydruk ma przykładową postać:

```
E9 FC DB E200 E103 E078
```

Kolejno wypisana jest: zawartość rejestru warunków **CC**, akumulatora **ACCA**, akumulatora **ACCB**, rejestru indeksowego **X**, licznika programu **PC** i wskaźnika stosu **SP**. Wartość SP = = $E078_H$ oznacza, że adresy pamięci RAM, w których pamiętana jest zawartość rejestrów są następujące: CC — $E079_H$; ACCA — $E07A_H$; ACCA — $E07B_H$; X — $E07C/E07D_H$; PC — $E07E/E07F_H$.

Przed wykonaniem dyrektywy C, N lub T zawartość stosu (a więc także rejestrów procesora w programie użytkownika) może być zmieniona dyrektywą M.

Każda próba zapisu do pamięci ROM lub pamięci nie istniejącej fizycznie sygnalizowana jest wydrukiem „?" i zakończeniem dyrektywy.

Dyrektywy programu MONITOR

Na każdą dyrektywę składa się jej kod (jedna litera i ewentualnie jeden lub dwa argumenty).

B — drukowanie adresów wszystkich zdefiniowanych pułapek programowych. W czasie wykonywania programu kod znajdujący się pod takim adresem zastępowany jest kodem instrukcji SWI.

D — skasowanie wszystkich pułapek.

C — wznowienie wykonywania programu po zatrzymaniu w pułapce. Kod instrukcji SWI zostaje zastąpiony oryginalnym kodem programu.

G xxxx — wykonanie programu od adresu xxxx.

M xxxx — operacja na zawartości komórki pamięci o adresie xxxx. Po przyjęciu dyrektywy, system drukuje zawartość podanej komórki w postaci szesnastkowej. Operator może teraz zażądać wydruku zawartości komórki o adresie xxxx+1, wprowadzając znak **LF**. Dla dokonania zmiany zawartości pamięci należy wpisać dwa znaki szesnastkowe i LF. Aby odczytać zawartość komórki o adresie xxxx—1, należy wprowadzić zamiast LF znak ↑ (ASCII:$5E_H$). Operacje na pamięci kończy wprowadzenie znaku różnego od: **LF**, ↑ i 0...9 i A...F. Może to być np. spacja lub kropka.

```
*M EC00 CF
EC01 9E 86
EC02 1D ^
EC01 86
EC02 1D 78
EC03 F7 .
```

Wydruk 1. Przykład operacji na zawartości pamięci

N — wykonanie jednego rozkazu po zatrzymaniu programu na pułapce. Drukowana jest aktualna zawartość wszystkich rejestrów procesora.

P xxxx yyyy — wydruk zawartości pamięci od adresu xxxx do adresu yyyy, w formacie S1-S9.

R — wydruk zawartości rejestrów.

T xxxx — podobnie jak dyrektywa C, z tym, że wykonaniu podanej szesnastkowo liczby rozkazów towarzyszy wydruk aktualnej zawartości rejestrów.

U xxxx — skasowanie pułapki pod adresem xxxx.

V xxxx — ustawienie pułapki pod adresem xxxx.

```
*D
*B
*V E000
*B E000
*V E050
*V E060
*B E000 E050 E060
*U E050
*B E000 E060
*D
*B
```

Wydruk 2. Ustawianie i kasowanie pułapek

```
*V E407
*B E407
*
*
*G
*G E400
CB 81 01 F000 E407 E078
*N
C0 81 02 F000 E408 E078
*N
CB 82 02 F000 E409 E078
*T 000A
CB 82 02 F001 E40A E078
CB 82 02 F001 E407 E078
C0 82 03 F001 E408 E078
CB 83 03 F001 E409 E078
CB 83 03 F002 E40A E078
CB 83 03 F002 E407 E078
C0 83 04 F002 E408 E078
CB 84 04 F002 E409 E078
CB 84 04 F003 E40A E078
CB 84 04 F003 E407 E078
*
```

Wydruk 3. Przykład użycia pułapki przy uruchamianiu programu

[1]) Program stanowi modyfikację firmowego monitora MINIBUG3E.

S xxxx yyyy — podobnie jak dyrektywa P, z tym, że wewnętrzny dzielnik ACIA ustawiany jest na podział częstotliwości TxC i RxC przez 64, wyjście RTS/ w stan „1", a przed wysłaniem żądanego bloku danych wysyłany jest ciąg 256 bajtów zerowych. Dyrektywa może być użyta do wyprowadzania zawartości pamięci na magnetofon lub perforator, przy czym szybkość transmisji wynosi 300 bodów, a sygnał RTS/ = „1" wskazuje, że dane nie są przeznaczone dla terminala.

```
*
*
*P FC00 FC0F
S113FC00FEE0006E00FEE0066E00FEE0346E00537F89
*
*
*
*
```

Wydruk 4. Przykładowa postać wydruku rekordu

L — ładowanie do pamięci blok zapisany komendą S. Wewnętrzny dzielnik i poziom sygnału RTS/ — jak wyżej.

Obsługa przerwań

Po przyjęciu przerwania adres odpowiedniej procedury obsługi pobierany jest sprzętowo z odpowiednich komórek w obszarze $FFF8_H$-$FFFD_H$. Aby jednak umożliwić obsługę przerwań, zgodnie z życzeniem użytkownika, procedury te oddają sterowanie pod adres pobrany z bufora w pamięci RAM. Jest on umieszczany w następujący sposób:

- adres obsługi IRQ/ : $E000/F001_H$
- adres obsługi NMI/ : $E006/E007_H$
- adres obsługi SWI/ : $E036/E037_H$

Adres dla SWI/ jest wstępnie wstawiany w trakcie inicjacji systemu tak, aby zawierał adres procedury obsługi w programie MONITOR.

Wydruk programu

Przedstawiony wydruk kodu maszynowego programu został wyprowadzony na drukarkę dyrektywą P. Wykonany został według formatu S1-S9 wprowadzonego przez firmę MOROTOLA. Blok danych dzielony jest na rekordy. Każdy rekord rozpoczyna się znakami początku S1. Następne dwa znaki określają liczbę bajtów w rekordzie (bez nich samych). Następne cztery znaki tworzą adres początku rekordu. Końcowe dwa znaki, to bajt sumy kontrolnej. Następne znaki (po dwa) oznaczają bajty danych. Koniec bloku oznaczany jest znakami S9. Cały blok jest zapisany w postaci ciągu znaków ASCII:0...9 i A...F oraz S. Przy wyprowadzaniu i wprowadzaniu dokonywana jest odpowiednia konwersja z (lub) na kod szesnastkowy. Przykładowy rekord przedstawiony jest na wydruku 4.

```
*
*
*P FC00 FFFF
S113FC00FEE0006E00FEE0066E00FEE0346E00537F
S113FC103904E078FED9FF0830037EFF467EFF4D8D
S113FC20BEE03FCEFC1FA60036098CFC1126F7C6D9
S113FC302A4F365A26FCB603D7B0088619B7E0197E
S113FC40BEE032B6E019B780087FE00CCEFD32DDCD
S113FC50FD18BDFD17BD7316BDFDB0CEFFD1E100EB
S113FC60270A0808088CFFF826FA20D4EE016E0059
S113FC708D5880302BCA81092F0A81112BC2B1161D
S113FC802EBE800739BDFD81FFE0028B4CFFE004EC
S113FC9039B6E01984FE8A42B780087CE00C8D7F77
S113FCA0B15326FA8D798139279A813124F07FE0B8
S113FCB00B8D348002B7E00E8D1F8D2B7AE00E2758
S113FCC00BA700A100260B0B20F020535C27CF864C
S113FCD007BD35863FBD312091BD0CB7E00F8D0750
S113FCE0B7E010FEE00F39BD874B4B4B4B16BDFC40
S113FCF0701B16FBE00DF7E00D3944444444840FB7
S113FD008B30B13923028B0737F6B008575724F943
S113FD10B7800933398DF108A600B10426F739B676
S113FD208008724FAB68009B47F7DE00C27D939FE
S113FD3020E6130A0D1400000000042A04BD42FE7C
S113FD40E00F8D3A098DD8810A2718815E270EBDF0
S113FD50FC728D95A700A10027EB7EFCCF860A8D4F
S113FD60A709850BFFE00FCEFD34BDACCEE00F8DE2
S113FD700B20CCA60008368D8132208B28DF58DF3C0
S113FD80BC8D60B62020B1CEFD338D8CFEE0320880
S113FD908DEC8DEA8DE88DE48DE2CEE03220D8D8D0
S113FDA0428609BC8628CB6112008BD368609BD21
S113FDB0378605BD337EFC40BD29FEE032A707B6D9
S113FDC0E00FA7067CE015CE00012009CE00018CCF
S113FDD0BD117FE015FFE00AFEE032EE06BDFF86DE
S113FDE07EFF337EFCD9868236CEE01AFFE01133E3
S113FDF037A602EE002703C4FB8CC4C47DE014279D
S113FE0003C4FD8CC4FE542402A70054240DA60090
S113FE1036863FA70032FEE011A702FEE01154240B
S113FE200EB6E00FA700B6E010A7013284FB3654EB
S113FE30240EA600B1E00F2607A601B1E0102703A7
S113FE405424046F006F01542404BDFD7C8C080B05
S113FE50088CE0322696F7E0143239BDFC85865AC8
S113FE60010101B7800BC6FF4FBDFD085A26F920DD
S113FE7003BDFCB5FEE002FFE011B6E005B0E01230
S113FE80F6E004F2E0112604B1102502860F8B04AB
S113FE90B7E00E8003B7E00DCEFED0BDFD185FCEF7
S113FEA0E00E8D27CEE0118D228D20FEE0118D1BFA
S113FEB07AE00D26F9FFE0115317BDFD7609BCE089
S113FEC00426B7CEFC0FBDFD182042EB007EFD7367
S113FED00D0A000000005331048FE032BD25306D5F
S113FEE0026026A056A06EE05BCE0082722A6007B
S113FEF0813F2614306C0626026C05BDFDE6FEE04B
S113FF00366E0086017EFDE8BDF9BDFD877EFC40DE
S113FF10B6E013A7007DE01527OD7FE015BDFDE6D3
S113FF207FE0087FE0093BBDFD87FEE00A09FFE0B2
S113FF300A27DAB6E013B7E038BD6E4D2A0A323458
S113FF408A10067EE03B8602FEE00BE600C16E27CD
S113FF5044C17E2750C1AD273CC1BD2748C13B27C2
S113FF603BC139273CC13F2713C1BD260DFEE00B54
S113FF70A6010B08Z0A04094C26FCBD158D08863F25
S113FF80A700BEE0323BFFE00BA600B7E013390B43
S113FF904A4D26FB39A60130EE0320DE30EE0C205C
S113FFA0DB30EE07BCEE0120D3164444444ACEFFEC
S113FFB0C18DDEA6002609B602C40DC10C26014CA3
S113FFC039010182010101020300020203000202SD
S113FFD00342FDA443FDC444FDA747FDB84CFC9176
S113FFE04DFD3D4EFDCC50FE7152FF0A53FE5B5455
S113FFF0FD0055FD9F56FDABFC00FC0AFC05FC2022S9
```

Wydruk 5. Kod maszynowy programu MONITOR

Procedury komunikacyjne MONITORA

Program komunikuje się z terminalem za pomocą następujących procedur:

INCHAR — procedura wprowadzająca znak z terminala. Wywołanie: **JSR INCHAR.** Funkcja: procedura czeka na gotowość sekcji odbiornika układu ACIA i ewentualnie wprowadza znak do ACCA. Bit D_7 jest zerowany i badana jest zawartość komórki $E00C_H$ (ECHO). Wartość zero pod tym adresem powoduje wywołanie procedury OUTCHAR. Wyjście: znak z terminala w ACCA. Adres procedury: $FD1F_H$.

OUTCHAR — procedura wyprowadza znak do terminala. Wejście: znak do wysłania w ACCA. Wywołanie: **JSR OUTCHAR.** Funkcja: procedura sprawdza gotowość sekcji nadajnika ACIA i ewentualnie wysyła znak do terminala. Adres procedury: $FD08_H$.

Obie procedury nie niszczą zawartości ACCB i X.

Struktura programu

Znajdujące się w programie tablice stałych są umiejscowione jak następuje: $FC0F$-$FC1F_H$, $FD32$-$FD3C_H$, $FED0$-$FED8_H$ i $FFC1$-$FFFF_H$.

Program MONITOR napisany jest w sposób niezwykle zwarty. Wykorzystane są wszystkie 1024 komórki pamięci. Stanowi to pewne utrudnienie przy wszelkich próbach rozszerzenia programu. Repertuar dyrektyw można jednak powiększyć umieszczając nową, bogatszą tablicę kodów i adresów poza obszarem programu. Należy wtedy zmienić adresy w procedurze przeszukiwania; nowy adres początkowy tabeli powinien być umieszczony pod adresem $FC5C/D_H$, a nowy adres końca tabeli (+ 1 komórka) w miejscu $FC66/7_H$.

Po kilku miejscach programu zastosowany jest pewien chwyt pozwalający na ścieśnienie kodu. Oto przykład:

```
FE01 C4 FD      ANDB # FD
FE03 8C C4FE    CPX # C4FE
FE06 54         LSRB
FE07 24 02      BCC DALEJ
```

* * *

Instrukcja ANDB # FD to iloczyn logiczny zawartości ACCB ze stałą FD_H. Program wykonany od adresu $FE01_H$ wykona operację iloczynu ze stałą FD_H, rozkaz porównania rejestru X ze stałą nie zmieni ACCB i wykonane zostaną rozkazy przesunięcia i skoku warunkowego. Jeżeli jednak z pewnego miejsca programu zostanie wykonany skok do adresu $FE04_H$ to pierwszy bajt a r g u m e n t u rozkazu CPX potraktowany zostanie jako kod rozkazu ANDB z argumentem FE_H. Wykonując zatem program od adresu $FE04_H$ rozkaz LSRB nastąpi po wykonaniu iloczynu ze stałą FE_H.

Powyższy zabieg pozwala na zaoszczędzenie jednej komórki pamięci programu.

JERZY ORKISZEWSKI
Warszawa

Wkładkę mikroKLAN prowadzi:

mgr inż. Andrzej J. Piotrowski
tel. domowy 48-22-85

Zanim wykręcisz ten numer weź pod uwagę, że jesteś jednym z kilkudziesięciu tysięcy zainteresowanych mikrokomputerami.